# Symetric

1. **using** System;
2. **using** System.Security.Cryptography;
3. **using** System.IO;
5. **namespace** SymmetricTutorial
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. // Create a new instance of the AES algorithm
12. SymmetricAlgorithm aes = **new** AesManaged();
14. **byte**[] key = aes.Key; // Key propery contains the key of the aes algorithm you can create your own
16. Console.WriteLine("Enter your message here to encrypt:");
17. **string** message = Console.ReadLine();
19. // Call the encryptText method to encrypt the a string and save the result to a file
20. EncryptText(aes, message, "encryptedData.dat");
22. // Call the decryptData method to get the encrypted text from the file and print it
23. Console.WriteLine("Decrypted message: {0}", DecryptData(aes, "encryptedData.dat"));


27. }
29. // Method to encrypte a string data and save it in a specific file using an AES algorithm
30. **static** **void** EncryptText(SymmetricAlgorithm aesAlgorithm,**string** text,**string** fileName)
31. {
32. // Create an encryptor from the AES algorithm instance and pass the aes algorithm key and inialiaztion vector to generate a new random sequence each time for the same text
33. ICryptoTransform encryptor = aesAlgorithm.CreateEncryptor(aesAlgorithm.Key, aesAlgorithm.IV);
35. // Create a memory stream to save the encrypted data in it
36. **using** (MemoryStream ms = **new** MemoryStream())
37. {
38. **using** (CryptoStream cs = **new** CryptoStream(ms,encryptor,CryptoStreamMode.Write))
39. {
40. **using** (StreamWriter writer = **new** StreamWriter(cs))
41. {
42. // Write the text in the stream writer
43. writer.Write(text);
44. }
45. }
47. // Get the result as a byte array from the memory stream
48. **byte**[] encryptedDataBuffer = ms.ToArray();
50. // Write the data to a file
51. File.WriteAllBytes(fileName, encryptedDataBuffer);
52. }
53. }
55. // Method to decrypt a data from a specific file and return the result as a string
56. **static** **string** DecryptData(SymmetricAlgorithm aesAlgorithm, **string** fileName)
57. {
58. // Create a decryptor from the aes algorithm
59. ICryptoTransform decryptor = aesAlgorithm.CreateDecryptor(aesAlgorithm.Key, aesAlgorithm.IV);
61. // Read the encrypted bytes from the file
62. **byte**[] encryptedDataBuffer = File.ReadAllBytes(fileName);
64. // Create a memorystream to write the decrypted data in it
65. **using** (MemoryStream ms = **new** MemoryStream(encryptedDataBuffer))
66. {
67. **using** (CryptoStream cs = **new** CryptoStream(ms, decryptor, CryptoStreamMode.Read))
68. {
69. **using** (StreamReader reader = **new** StreamReader(cs))
70. {
71. // Reutrn all the data from the streamreader
72. **return** reader.ReadToEnd();
73. }
74. }
75. }
76. }
77. }
78. }

# Assymetric

1. **using** System;
2. **using** System.IO;
3. **using** System.Security.Cryptography;
4. **using** System.Text;
6. **namespace** AsymmetricTutorial
7. {
8. **class** Program
9. {
11. **static** **void** Main(**string**[] args)
12. {
13. // Create an instance of the RSA algorithm class
14. RSACryptoServiceProvider rsa = **new** RSACryptoServiceProvider();
15. // Get the public keyy
16. **string** publicKey = rsa.ToXmlString(**false**); // false to get the public key
17. **string** privateKey = rsa.ToXmlString(**true**); // true to get the private key
19. // Call the encryptText method
20. EncryptText(publicKey, "Hello from C# Corner", "encryptedData.dat");
22. // Call the decryptData method and print the result on the screen
23. Console.WriteLine("Decrypted message: {0}", DecryptData(privateKey, "encryptedData.dat"));
25. }
27. // Create a method to encrypt a text and save it to a specific file using a RSA algorithm public key
28. **static** **void** EncryptText(**string** publicKey ,**string** text,**string** fileName)
29. {
30. // Convert the text to an array of bytes
31. UnicodeEncoding byteConverter = **new** UnicodeEncoding();
32. **byte**[] dataToEncrypt = byteConverter.GetBytes(text);
34. // Create a byte array to store the encrypted data in it
35. **byte**[] encryptedData;
36. **using** (RSACryptoServiceProvider rsa = **new** RSACryptoServiceProvider())
37. {
38. // Set the rsa pulic key
39. rsa.FromXmlString(publicKey);
41. // Encrypt the data and store it in the encyptedData Array
42. encryptedData = rsa.Encrypt(dataToEncrypt, **false**);
43. }
44. // Save the encypted data array into a file
45. File.WriteAllBytes(fileName, encryptedData);
47. Console.WriteLine("Data has been encrypted");
48. }
50. // Method to decrypt the data withing a specific file using a RSA algorithm private key
51. **static** **string** DecryptData(**string** privateKey,**string** fileName)
52. {
53. // read the encrypted bytes from the file
54. **byte**[] dataToDecrypt = File.ReadAllBytes(fileName);
56. // Create an array to store the decrypted data in it
57. **byte**[] decryptedData;
58. **using** (RSACryptoServiceProvider rsa = **new** RSACryptoServiceProvider())
59. {
60. // Set the private key of the algorithm
61. rsa.FromXmlString(privateKey);
62. decryptedData = rsa.Decrypt(dataToDecrypt, **false**);
63. }
65. // Get the string value from the decryptedData byte array
66. UnicodeEncoding byteConverter = **new** UnicodeEncoding();
67. **return** byteConverter.GetString(decryptedData);
68. }
69. }
70. }